

## EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S3	3545645	@ad<"20011019"	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:07
L2	20	l1 and (agent or bot) with parse with query	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:08
L1	3867337	@ad<"20011019" or @rlad<"20011019"	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:08
L6	69	l1 and (agent or bot) with parse same database	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:29
L5	173	l1 and (agent or bot) with parse and (agent or bot) with database	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:29
L4	0	l1 and (agent or bot) with parse with sentence same query and (agent or bot) with database	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:29
L3	11	l1 and (agent or bot) with parse with query and (agent or bot) with database	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:29
L8	1	l1 and (agent or bot or robot) with parse same search with database and 709/202.ccls.	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:34
L7	4	l1 and (agent or bot) with parse same database and 709/202.ccls.	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:34
L11	45	("6009422").URPN.	USPAT	AND	ON	2007/07/25 09:43
L10	6	("5408652"   "5416917"   "5590319"   "5596744"   "5634053"   "5655116").PN.	US-PGPUB ; USPAT; USOCR	AND	ON	2007/07/25 09:43
L13	33	("5655116").URPN.	USPAT	AND	ON	2007/07/25 09:49

# Internet Relay Chat

From Wikipedia, the free encyclopedia

**Internet Relay Chat (IRC)** is a form of real-time Internet chat or synchronous conferencing. It is mainly designed for group (many-to-many) communication in discussion forums called *channels*, but also allows one-to-one communication and data transfers via private message.

IRC was created by Jarkko "WiZ" Oikarinen in late August 1988 to replace a program called MUT (MultiUser talk) on a BBS called OuluBox in Finland. Oikarinen found inspiration in a chat system known as Bitnet Relay, which operated on the BITNET.

IRC gained prominence when it was used to report on the Soviet coup attempt of 1991 throughout a media blackout. It was previously used in a similar fashion by Kuwaitis during the Iraqi invasion. Relevant logs are available from ibiblio

(<http://www.ibiblio.org/pub/academic/communications/logs/>) archive.

IRC client software is available for virtually every computer operating system.

## The five-layer TCP/IP model

### 5. Application layer

DHCP • DNS • FTP • Gopher • HTTP • IMAP4 • IRC • NNTP • XMPP • POP3 • SIP • SMTP • SNMP • SSH • TELNET • RPC • RTP • RTCP • RTSP • TLS/SSL • SDP • SOAP • VPN • PPTP • L2TP • GTP • STUN • NTP • ...

### 4. Transport layer

TCP • UDP • DCCP • SCTP • ...

### 3. Internet Layer

IP (IPv4 • IPv6) • IGMP • ICMP • RSVP • BGP • OSPF • ISIS • IPsec • ARP • RARP • RIP • ...

### 2. Data link layer

802.11 • ATM • DTM • Ethernet • FDDI • Frame Relay • GPRS • EVDO • HSPA • HDLC • PPP • ...

### 1. Physical layer

Ethernet physical layer • ISDN • Modems • PLC • SONET/SDH • G.709 • WiMAX • ...

## Contents

- 1 Technical information
  - 1.1 Commands and replies
  - 1.2 Channels
  - 1.3 Modes
  - 1.4 IRC operators
  - 1.5 Attacks
  - 1.6 Abuse prevention: Timestamping vs. nick/channel delay protocol
    - 1.6.1 Nick/channel delay
    - 1.6.2 Timestamping
- 2 Networks and URLs
- 3 Clients
  - 3.1 Bots
  - 3.2 Bouncer

## EAST Search History

L12	18	("4408302"   "4769772"   "4933800"   "5021989"   "5117349"   "5175814"   "5212787"   "5226111"   "5241621"   "5263126"   "5265014"   "5315703"   "5355320"   "5355474"   "5369761"   "5379366"   "5388196"   "5408655").PN.	US-PGPUB ; USPAT; USOCR	AND	ON	2007/07/25 09:49
L9	19	l1 and (agent or bot or robot) with parse same (search or query) with database	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:52
L14	6	l1 and (agent or bot or robot) and parse with (sentence or phrase) same (search or query) with database	US-PGPUB ; USPAT	AND	ON	2007/07/25 09:53

- 4 Search engines
- 5 Modern IRC
- 6 File sharing
- 7 See also
- 8 References
- 9 External links

## Technical information

IRC is an open protocol that uses TCP and optionally SSL. An IRC server can connect to other IRC servers to expand the IRC network. Users access IRC networks by connecting a client to a server. There are many client and server implementations, such as mIRC and the Bahamut IRCd. Most IRC servers do not require users to log in, but a user will have to set a nickname before being connected.

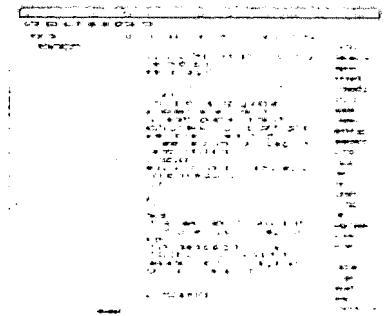
IRC was originally a plain text protocol (although later extended),<sup>[1]</sup> which on request was assigned port 194/TCP by IANA; although, the de facto has always been running on 6667/TCP and nearby port numbers to avoid having to run the IRCd software with root privileges. It is possible (though quite inconvenient) to use IRC via a basic byte-stream client such as netcat or telnet.

The protocol specified that characters were 8 bit but did not specify the character encoding the text was supposed to use. In the days when IRC was introduced people used text terminals that only supported one encoding at a time and presumably the designers assumed that those who wanted to converse would use sufficiently compatible encodings. This can cause problems when users using different clients and/or different platforms want to converse in languages other than English.

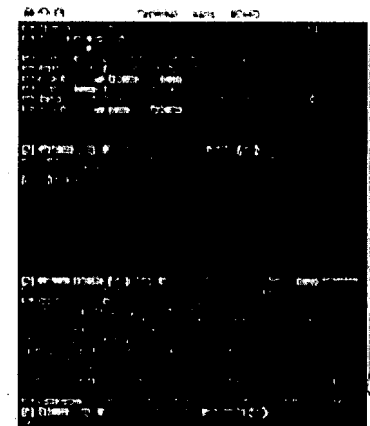
All client-to-server IRC protocols in use today are descended from the protocol implemented in the irc2.8 version of the IRC2server, and documented in RFC 1459 (<http://tools.ietf.org/html/rfc1459>). Since RFC 1459 (<http://tools.ietf.org/html/rfc1459>) was published, the new features in the irc2.10 implementation led to the publication of several revised protocol documents; RFC 2810 (<http://tools.ietf.org/html/rfc2810>), RFC 2811 (<http://tools.ietf.org/html/rfc2811>), RFC 2812 (<http://tools.ietf.org/html/rfc2812>) and RFC 2813 (<http://tools.ietf.org/html/rfc2813>), however these protocol changes have not been widely adopted among other implementations.

Although many specifications on the IRC protocol have been published, there is no official specification, as the protocol remains dynamic. Virtually no clients and very few servers rely strictly on the above RFCs as a reference.

The IRC protocol was customized by Microsoft in 1998 via its proprietary IRCX protocol that addresses



A Screenshot of Xchat, an IRC client.



Xaric, a console based IRC client, in use on Mac OS X. Shown are two IRC channels and a private conversation with the software author.

some of the traditional issues that legacy IRC networks faced, along with some features that most users felt were 'ahead of its time'.

The standard structure of a network of IRC servers is a tree (acyclic graph). Messages are routed along only necessary branches of the tree but network state is sent to every server and there is generally a high degree of implicit trust between servers. This architecture has a number of problems; a misbehaving or malicious server can cause major damage to the network, any changes in structure whether intentional or a result of conditions on the underlying network require a net-split and net-join resulting in a lot of network traffic and spurious quit/join messages to users and temporary loss of communication and adding a server to a large network means a large background bandwidth load on the network and a large memory load on the server.

For a discussion of the evolution of server-side IRC protocols and the various IRCd incarnations, please see the separate article on IRC daemons.

## Commands and replies

IRC is based on a line-based structure with the client sending single-line messages to the server, receiving replies to those messages and receiving copies of some messages sent by other clients. In most clients users can enter commands by prefixing them with `/`; depending on the command, these may either be passed directly to the server (generally used for commands the client does not recognise), passed to the server with some modification or handled entirely by the client.

## Channels

The basic means of communication in an established IRC session is a *channel*. You can see all the channels in a server using the command `/list [#string] [-min #] [-max #]` that lists all currently available channels, optionally filtering for parameters (`#string` for the entire or part of the name, with wildcards, and `#min` / `#max` for number of users in the channel).

Users can *join* to a channel using the command `/join #channelname` and send messages to it, which are relayed to all other users on the same channel.

Channels that are available across an entire IRC network are prepended with a `#`, while those local to a server use `&`. Other non-standard and less common channel types include `+` channels — 'modeless' channels without operators, and `!` channels, a form of timestamped channel on normally non-timestamped networks.

## Modes

Users and channels may have *modes*, which are represented by single case-sensitive letters and are set using the mode command. User modes and channel modes are separate and can use the same letter to mean different things (e.g. usermode `"i"` is invisible mode whilst channelmode `"i"` is invite only). Modes are usually set and unset using the mode command which takes a target (user or channel), a set of modes to set (+) or unset (-) and any parameters the modes need.

Some but not all channel modes take parameters and some channel modes apply to a user on a channel or add or remove a mask (e.g. a ban mask) from a list associated with the channel rather than applying to

the channel as a whole. Modes that apply to users on a channel have an associated symbol which is used to represent the mode in names replies (sent to clients on first joining a channel and use of the names command) and in most clients to represent it in this list of users in the channel.

In order to correctly parse incoming mode messages and track channel state the client must know which mode is of which type and for the modes that apply to a user on a channel which symbol goes with which letter. In early implementations of IRC this had to be hard-coded in the client but there is now a de-facto standard extension to the protocol which sends this information to the client at connect time.

There is a small design fault in IRC regarding modes that apply to users on channels, the names message used to establish initial channel state can only send one such mode per user on the channel, but multiple such modes can be set on a single user. For example, if a user is holds both operator status (+o) and voice status (+v) on a channel, a new client will be unable to know the less precedented mode (voice). Workarounds for this are possible on both the client and server side but none are widely implemented.

Standard (rfc1459) modes					
User modes		Channel modes			
Letter	Description	Letter	Symbol	Parameter	Description
<b>i</b>	Invisible — cannot be seen without a common channel or knowing the exact name	<b>o</b>	@	Name of affected user	Channel operator — can change channel modes and kick users out of the channel among other things
<b>s</b>	Receives server notices	<b>p</b>	None	None	Private channel — listed in channel list as "prv" according to rfc1459
<b>w</b>	Receives wallops	<b>s</b>	None	None	Secret channel — not shown in channel list or user whois except to users already on the channel
<b>o</b>	User is an IRC operator (ircop)	<b>i</b>	None	None	Invite only — users can only join if invited by a channel operator
		<b>t</b>	None	None	Topic only settable by channel operators
		<b>n</b>	None	None	Users not on the channel cannot send messages to it
		<b>m</b>	None	None	Channel is moderated (only those who hold operator or voice status on the channel can send messages to it)
		<b>l</b>	None	Limit number	Limits number of users able to be on channel (when full, no new users can join)
				Ban mask (nick!	

<b>b</b>	None	user@host with wildcards allowed)	Bans hostmasks from channel
<b>v</b>	+	Name of affected user	Gives a user voice status on channel (see +m above)
<b>k</b>	None	New channel key	Sets a channel key such that only users knowing the key can enter

Many ircd coders have added extra modes or modified the behavior of modes in the above list so it is strongly advisable to check the documentation of the irc network or ircd (though note that the network may have patched the ircd) for more detailed information on what the modes do on a particular server or network.

## IRC operators

There are also users whose privileges extend to whole servers or networks of servers; these are called IRC Operators. On some IRC implementations, IRC operators are also given channel operator status in every channel, although many people believe that administration of channels and administration of the network should be kept separate, and that IRC operator status does not confer the right to interfere with a particular channel's operation.

## Attacks

Because IRC connections are usually unencrypted and typically span long time periods, they are an attractive target for malicious hackers. Because of this, careful security policy is necessary to ensure that an IRC network is not susceptible to an attack such as an IRC takeover war. IRC networks may also k-line or g-line users or networks that have a harming effect.

A small number of IRC servers support SSL connections for security purposes. This helps stop the use of packet sniffer programs to obtain the passwords of IRC users, but has little use beyond this scope due to the public nature of IRC channels. SSL connections require both client and server support (which may require the user to install SSL binaries and IRC client specific patches or modules on their computers).

IRC served as an early laboratory for many kinds of Internet attacks, such as using fake ICMP unreachable messages to break TCP-based IRC connections ("nuking") to annoy users or facilitate takeovers.

## Abuse prevention: Timestamping vs. nick/channel delay protocol

One of the most contentious technical issues surrounding IRC implementations, which survives to this day, is the merit of "Nick/Channel Delay" vs. "TimeStamp" protocols. Both methods exist to solve the problem of denial-of-service attacks, but take very different approaches.

The problem with the original IRC protocol as implemented was that when two servers split and rejoined, the two sides of the network would simply merge their channels. If a user could join on a

"split" server, where a channel which existed on the other side of the network was empty, and gain operator status, they would become a channel operator of the "combined" channel after the netsplit ended; if a user took a nickname which existed on the other side of the network, the server would kill both users when rejoining.

This was often abused to "mass-kill" all users on a channel, thus creating "opless" channels where no operators were present to deal with abuse. Apart from causing problems within IRC, this encouraged people to conduct denial of service attacks against IRC servers in order to cause netsplits, which they would then abuse.

### Nick/channel delay

The nick/channel delay (abbreviated ND/CD) solution to this problem was very simple. After a user signed off and the nickname became available, or a channel ceased to exist because all its users left (as often happens during a netsplit), the server would not allow any user to use that nickname or join that channel, respectively, until a certain period of time (the *delay*) had passed. The idea behind this was that even if a netsplit occurred, it was useless to an abuser because they could not take the nickname or gain operator status on a channel, and thus no collision of a nickname or 'merging' of a channel could occur. To some extent, this inconvenienced legitimate users, who might be forced to briefly use a different name (appending an underscore was popular) after rejoining.

### Timestamping

The alternative, the timestamp or *TS* protocol, took a different approach. Every nickname and channel on the network was assigned a timestamp -- the date and time when it was created. When a netsplit occurred, two users on each side were free to use the same nickname or channel, but when the two sides were joined, only one could survive. In the case of nicknames, the newer user, according to their TS, was killed; when a channel collided, the members (users on the channel) were merged, but the channel operators on the "losing" side of the split were de-oped.

TS is a much more complicated protocol than ND/CD, both in design and implementation, and despite having gone through several revisions, some implementations still have problems with "desyncs" (where two servers on the same network disagree about the current state of the network), and allowing too much leniency in what was allowed by the 'losing' side. Under the original TS protocols, for example, there was no protection against users setting bans or other modes in the losing channel which would then be merged when the split rejoined, even though the users who had set those modes were no longer opped. Some modern TS-based IRC servers have also incorporated some form of ND and/or CD in addition to timestamping in an attempt to further curb abuse.

There is not, and likely never will be, a consensus on timestamping vs. delay; however most networks today use the timestamping approach. It was part of the issues and disagreements which caused several servers to split away from EFnet and form the newer IRCnet (EFnet after the split moving to a TS protocol, and IRCnet using ND/CD), and supporters on both sides were known for heated arguments regarding the merits of their solution.

## Networks and URLs

There are thousands of running IRC networks in the world. They run various implementations of IRC

servers, and are administered by various groups of IRC operators, but the protocol exposed to IRC users is very similar, and all IRC networks can be accessed by the same client software.

One can join servers by clicking on a `irc://irc.server.net:port/channel` web link.

The largest IRC networks have traditionally been grouped in *The Big Four* — a designation for networks that top the statistics. The Big Four networks change periodically, but due to the community nature of IRC there are a large number of other networks for users to choose from.

### The Big Four:

- EFnet
- IRCnet
- QuakeNet
- Undernet

For network statistics, rankings, and a list of smaller networks, see [netsplit.de](http://netsplit.de) (<http://irc.netsplit.de/networks/>), Search IRC (<http://searchirc.com/networks>) and Gogloom (<http://gogloom.com/networks>). For other articles on IRC networks, see [Category:IRC networks](#).

### IRC networks

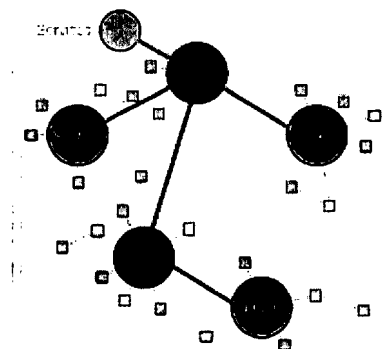
[hide]

Abjects • AbleNET • AfterNET • AusIRC • AustIRC • AustNet • Blitzed • BRASnet • DALnet • Delinked • EFnet • EsperNet • freenode • GamesNET • GameSurge • Global Gamers Center • IRC-Hispano • IRCHighway • IRCnet • NetGamers • OFTC • QuakeNet • Rizon • RusNet • SlashNET • UniBG • Undernet • ZiRC

## Clients

*See list of IRC clients for more detail.*

To connect to an IRC network, people most commonly will connect to a IRC network via a IRC client. The client takes the raw IRC traffic and turns it into a easy-to-use interface, removing the need for the user to know the IRC commands specified in RFC 1459 (<http://www.faqs.org/rfcs/rfc1459.html>), and later in RFC 2812 (<http://tools.ietf.org/html/rfc2812>). The client also simplifies some commands, like turning PRIVMSG into MSG, and also allowing the user to talk more freely. For example, instead of a user having to type the entire line "PRIVMSG #Wikipedia :Hey guys!" the user would only need to type into the channel window "Hey Guys!". Another example is a user identify to a IRC service. Instead of typing "PRIVMSG NickServ IDENTIFY ilikewikipedia", the user types "/AUTH ilikewikipedia". IRC clients also will convert some of the raw data into windows, like turning channel traffic and the user list sent when a user first enters a channel into two windows.



Scheme of an IRC-Network with normal clients (green), bots (blue) and bouncers (orange)

There are a great deal of IRC Clients available, and are mostly separated by Operating System. On Windows-based systems, one of the most popular IRC clients is mIRC.<sup>[2]</sup> However, with the recent introduction of new clients such as Bersirc, KVIrc, Trillian, Visual IRC, and X-Chat, mIRC is beginning to see much more competition, especially with clients that are not commercial. Many people still use mIRC most likely due to the fact that it has been around for quite some time and has a wide variety of scripts available.

ircII is the canonical Unix IRC client, but its userbase has declined with the appearance of competing clients such as ircII-EPIC, BitchX, irssi, X-Chat, Konversation, etc. For Mac OS X, the most widely used clients are Snak, Ircle and Colloquy. OS X can also run most Unix-like command line and X11 IRC clients. Recently, X-Chat Aqua, a special build of X-Chat, has been gaining ground on OS X systems.

Some Internet browsers have IRC Clients. Mozilla Firefox has an IRC client, ChatZilla, along with Opera, which also has a IRC client, but the client is built into it.

Some computer games have built-in IRC Clients, such as Unreal Tournament and Uplink.

For a novice user, mIRC and other large-window clients might seem to be unnecessarily large and complex. New users may prefer instant messaging clients like Miranda IM or Trillian, providing a familiar interface to the IRC application. The multi-platform open-source instant messaging client Pidgin also supports connection to the IRC networks.

## **Bots**

There are also many automated clients, called bots. The first bot was written by Greg Lindahl and provided moderation for the game of Hunt the Wumpus. As bots evolved, they began to serve as permanent points of contact for information exchange and protection agents for the channels they served, because of their superior speed when compared to humans. Presently, although many of these functions are often delegated to network-provided services which allow for registration and management of both nicknames and channels, bots still remain popular and continue to be adapted to new and unexpected tasks.

Bots have been written in a variety of languages, and a wide array of implementations exist, although recently and partly due to the overwhelming popularity of the mIRC IRC client, an increasing number of bots are written using the mIRC scripting language and run inside the client. Two popular<sup>[3]</sup> IRC bots are Eggdrop and EnergyMech, both written in the C programming language with optional add ins written in TCL or other scripting languages.

Most modern IRC services typically implement bot-like interfaces, through which users can communicate with and control the functionality.

Some bots were created for malevolent uses, such as flooding or taking over channels, occupying them from rightful owner(s). Malicious bots have been restricted in recent years.

## **Bouncer**

A program that runs as a daemon on a server and functions as a persistent proxy is known as a bouncer.

A bouncer's purpose is to maintain a connection to an IRC server, acting as a relay between it and the connecting client. Should the client lose network connectivity, the bouncer will archive all traffic for later delivery, allowing the user to resume his IRC session without externally perceptible disruption. Two of the most popular bouncers are muh (<http://mind.riot.org/muh/>) and psyBNC. Muh is exclusively for single user connections, while psyBNC supports multiple users. Another feature-rich bouncer is ZNC (<http://znc.sourceforge.net/>).

Furthermore, as a way of obtaining a bouncer-like effect, the old UNIX user's way of doing this is to run a (typically text-based) client on a remote server, inside a piece of screen-detaching software (e.g. GNU Screen), and using a secure shell to connect to this server, letting it relay all information, and thus letting the client archive all traffic should the connectivity be lost.

## Search engines

There are numerous search engines available to aid the user in finding what they are looking for on IRC. Generally the search engine consists of two parts. First being the "back-end" or the "spider/crawler". This is the work horse of the search engine. It is responsible for crawling IRC servers to index the information being sent across them. The information that is indexed usually consists solely of channel text (text that is publicly displayed in public channels). The storage method is usually some sort of relational database, like MySQL or Oracle. Which database that is used depends exclusively on the programming language in which the spider bot is written. The second part to the equation is the front-end "search engine". This search engine is the user interface to the database. It supplies the user with a way to search the database of indexed information to retrieve the data they are looking for. These front-end search engines can also be coded in numerous programming languages. The more popular languages for such search engines and indexing spiders are: Perl, PHP and C. Most search engines have their own spider that is a single application that is responsible for crawling IRC and indexing data itself; however, others are "user based" indexers.

What this means is that they rely on users to install their "add-on" to their IRC client (like mIRC) and this add-on is what sends the database the channel information of whatever channel[s] the user happens to be on. IRC search engines have completely automated the process of finding information on IRC and have thus contributed greatly to the popularity of IRC in recent years.

## Modern IRC

IRC has changed much over its life on the Internet. New server software has added a multitude of new features.

- **Services:** Network-operated bots to facilitate registration of nicknames and channels, sending messages for offline users and network operator functions.
- **Extra Modes:** While the original IRC system used a set of standard user and channel modes, new servers add many new modes for such features as removing color codes from text, or obscuring a user's hostmask ("cloaking") to protect from denial of service attacks.
- **Proxy Detection:** Most modern servers support detection of users attempting to connect through an insecure (misconfigured or exploited) proxy server, which can then be denied a connection. An example is the Blitzed Open Proxy Monitor (<http://www.blitzed.org/proxy>) or BOPM, used by several networks.
- **Additional Commands:** New commands can be such things as shorthand commands to issue

commands to Services, to network operator only commands to manipulate a user's hostmask.

- Encryption: For the client-to-server leg of the connection SSL might be used (messages cease to be secure once they are relayed to other users on standard connections, but it makes eavesdropping on or wiretapping an individual's IRC sessions difficult). For client-to-client communication, SDCC (Secure DCC) can be used.
- Ident: Provides identification to the IRC server.
- Connection Protocol: IRC can be connected to via IPv4, the current standard version of the Internet Protocol, or by IPv6, the next-generation version of the Protocol.

## File sharing

Using scripts like Sysreset (<http://www.sysreset.com/>), UPP (<http://upp.monkey-pirate.com/>), Polaris (<http://www.polaris-central.com/>) and, most commonly, OmenServe, users can create file servers that allow them to share files with others. In addition to the normal pros and cons of file-sharing (see Copyright infringement of software), there are also groups that set up anime fansubbing networks, allowing American audiences to see anime that would normally be unavailable in English or outside of Japan.

Due to the large number of people who use IRC for file sharing, some think of IRC as a form of P2P file sharing. Conversely, many users try to defeat this view by persistently discouraging it or refusing to help with it. Technically, IRC provides no file transfer mechanisms itself; file sharing is implemented by IRC *clients*, typically using the Direct Client-to-Client (DCC) protocol, in which file transfers are negotiated through the exchange of private messages between clients. The vast majority of IRC clients feature support for DCC file transfers, hence the view that file sharing is an integral feature of IRC.

## See also

- Comparison of instant messaging protocols
- Comparison of IRC clients
- Comparison of IRC daemons
- Comparison of IRC services
- List of IRC commands
- Bash.org
- Chat room
- Chatzilla
- Direct Client-to-Client
- Ident
- Instant messaging
- IRC Services
- IRCd
- IRCX
- Irssi
- mIRC
- psyBNC
- Serving channel
- XDCC

## References

1. ^ IRC RFC (<http://www.irchelp.org/irchelp/rfc/rfc.html>)
2. ^ Schweitzer, Douglas, *Internet Relay Chat: IRC the Place to Be?* (<http://www.pcflank.com/art32.htm>). Retrieved on 2007-03-25
3. ^ <http://uptime.eggheads.org/stats/>

## External links

- RFC 1459 (<http://tools.ietf.org/html/rfc1459>) - Technical Information about the IRC Protocol
- Large archive of IRC-related documents, somewhat EFNet biased (<http://www.irchelp.org/>)
- IRC.org - Technical and Historical IRC6 information; articles on the history of IRC. (<http://www.irc.org/>)
- Living Internet (<http://www.livinginternet.com/r/r.htm>) A comprehensive history of the Internet, including IRC.

Retrieved from "[http://en.wikipedia.org/wiki/Internet\\_Relay\\_Chat](http://en.wikipedia.org/wiki/Internet_Relay_Chat)"

Categories: Articles with unsourced statements since July 2007 | All articles with unsourced statements | IRC | Virtual communities | On-line chat

- 
- This page was last modified 04:35, 11 July 2007.
  - All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.) Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a US-registered 501(c)(3) tax-deductible nonprofit charity.